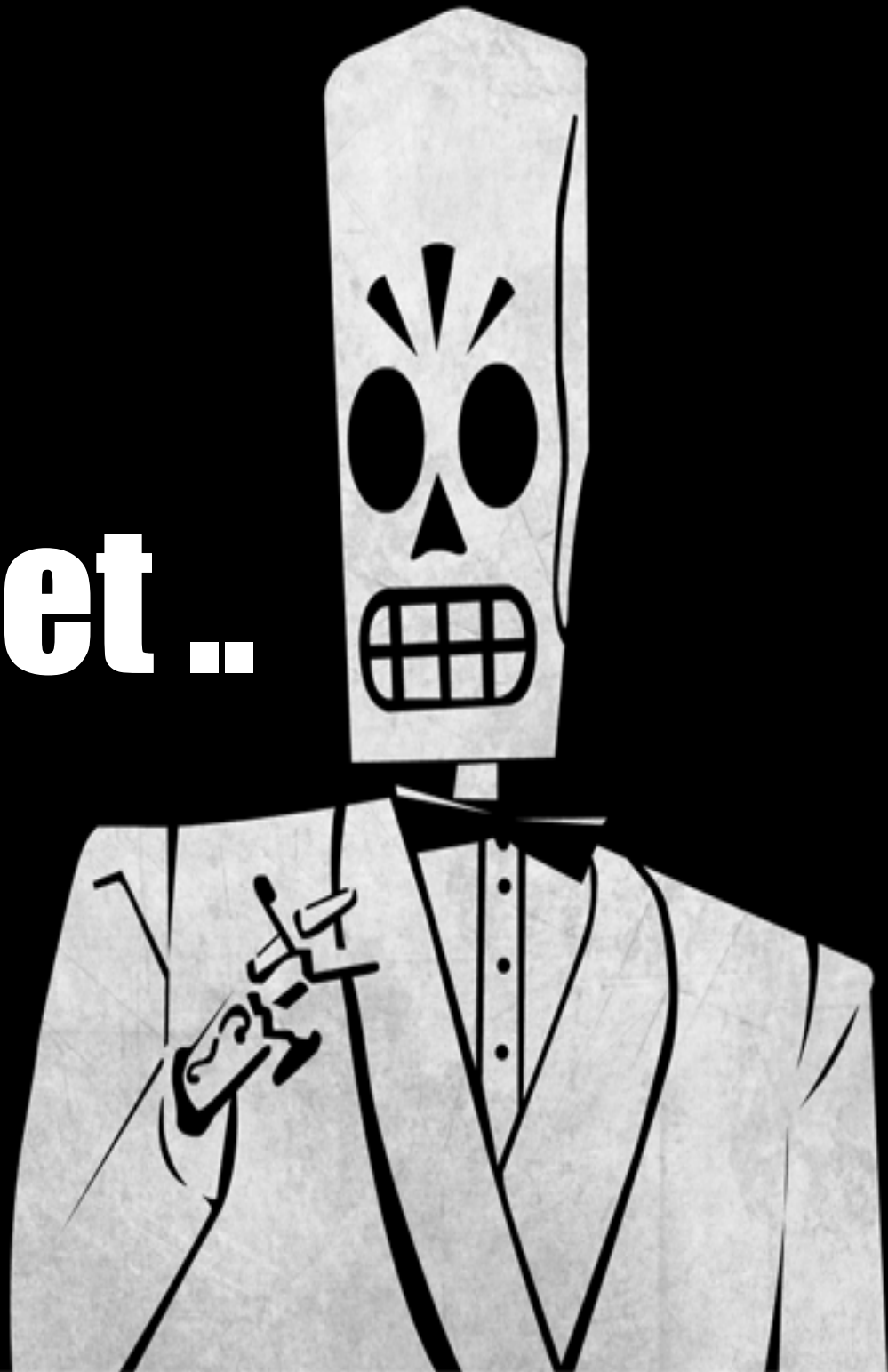
Die socket ..
Die !



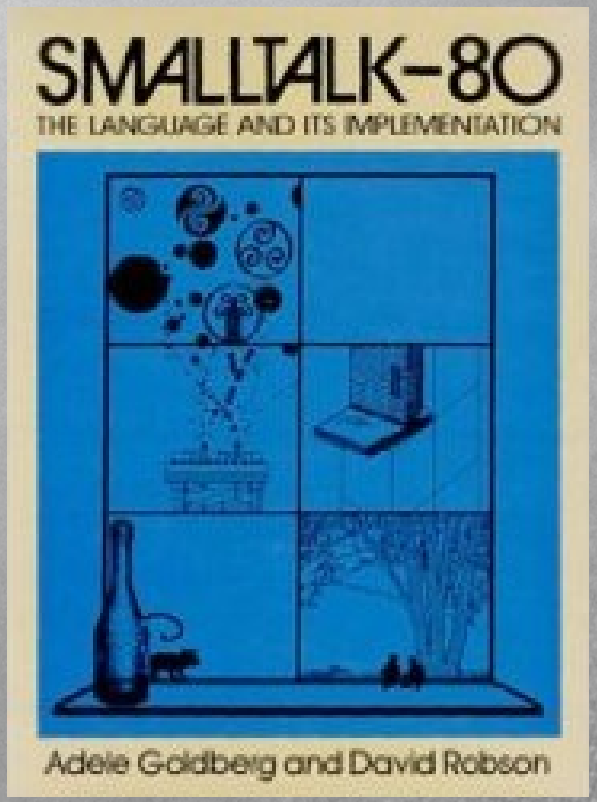
eamless
Let a thousand systems.. bloom !
a short story in 4 parts

PART ONE

DEATH TO SOCKETS

FOR THE

OF



STOP USING SOCKETS

```
get: limit dataInto: dataStream
    "Reel in data until the server closes the connection or the limit is reached.
    At the same time, watch for errors on otherSocket."

    | buf bytesRead currentlyRead |
    currentlyRead := 0.
    buf := String new: 4000.
    [currentlyRead < limit and:
    [self dataSocket isConnected or: [self dataSocket dataAv
        whileTrue: [
            self checkForPendingError:
            bytesRead := self dataSocke  at  h  outIn  uf.
            1 to: (bytesRead min: (limit - c  d))  dataStream nextPut: (buf at: ii)].
            currentlyRead := currentlyRead  ytesRead].
    dataStream reset.   "position: 0."
    ^ dataStream
```

DO NOT !

transcript

```
"self transcript"

(1 to: 100) do: [i |
    (Transcript from: '127.0.0.1:8081') show: i; cr.
].
```

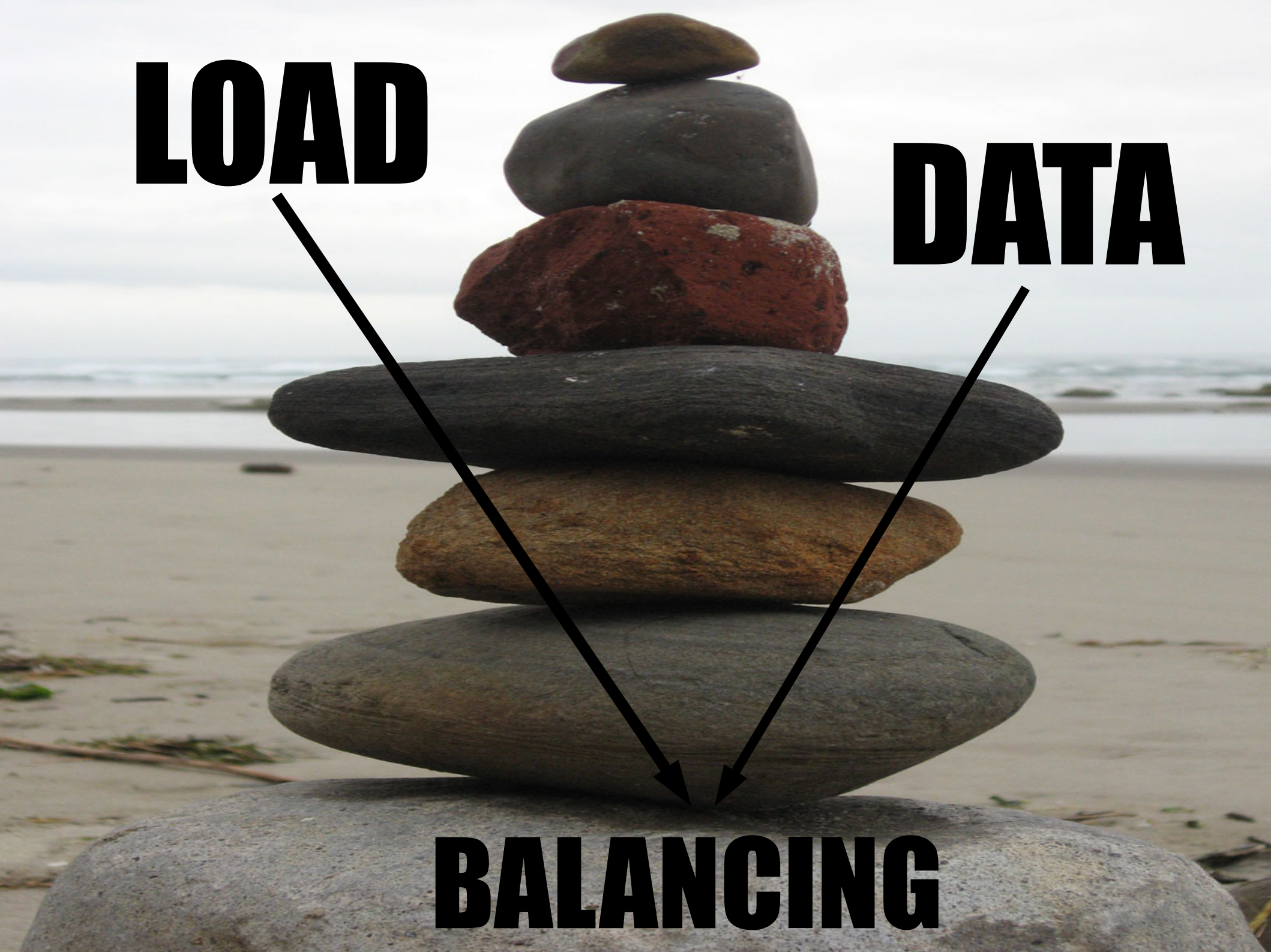
NO SILLY EXAMPLES

PART TWO

PROGRAM LOCALLY DEPLOY GLOBALLY

The x % of the applications that you write

Every time you give your model directly to the UI..


LOAD
DATA
BALANCING

INTEGRITY ?

PART THREE

LET A THOUSAND SYSTEMS BLOOM

THERE IS
A
PROBLEM

2 SOLUTIONS POWERED BY SEAMLESS

HEAD-FIRST ATTACK

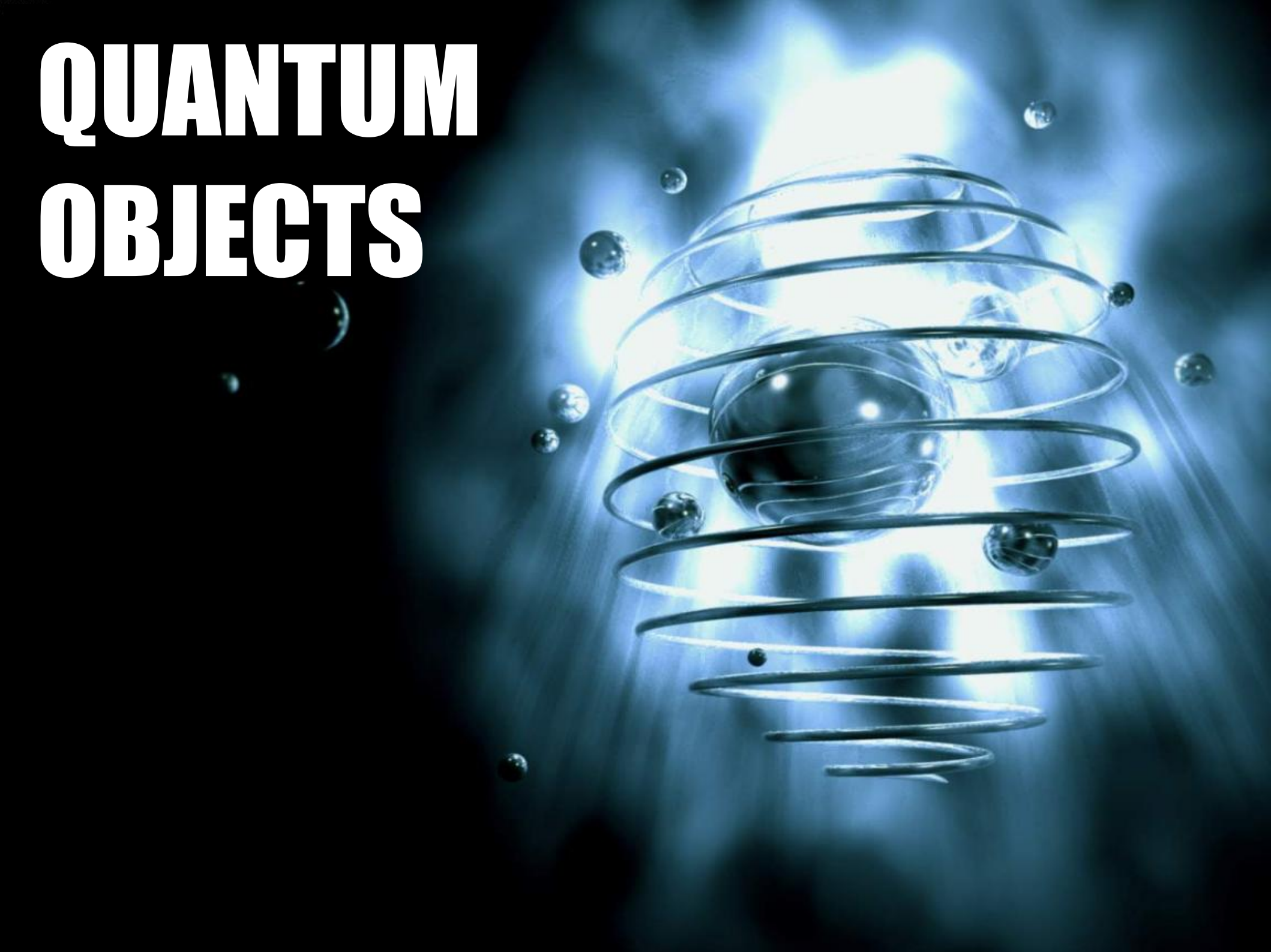
DISTRIBUTION AWARE META-OBJECTS

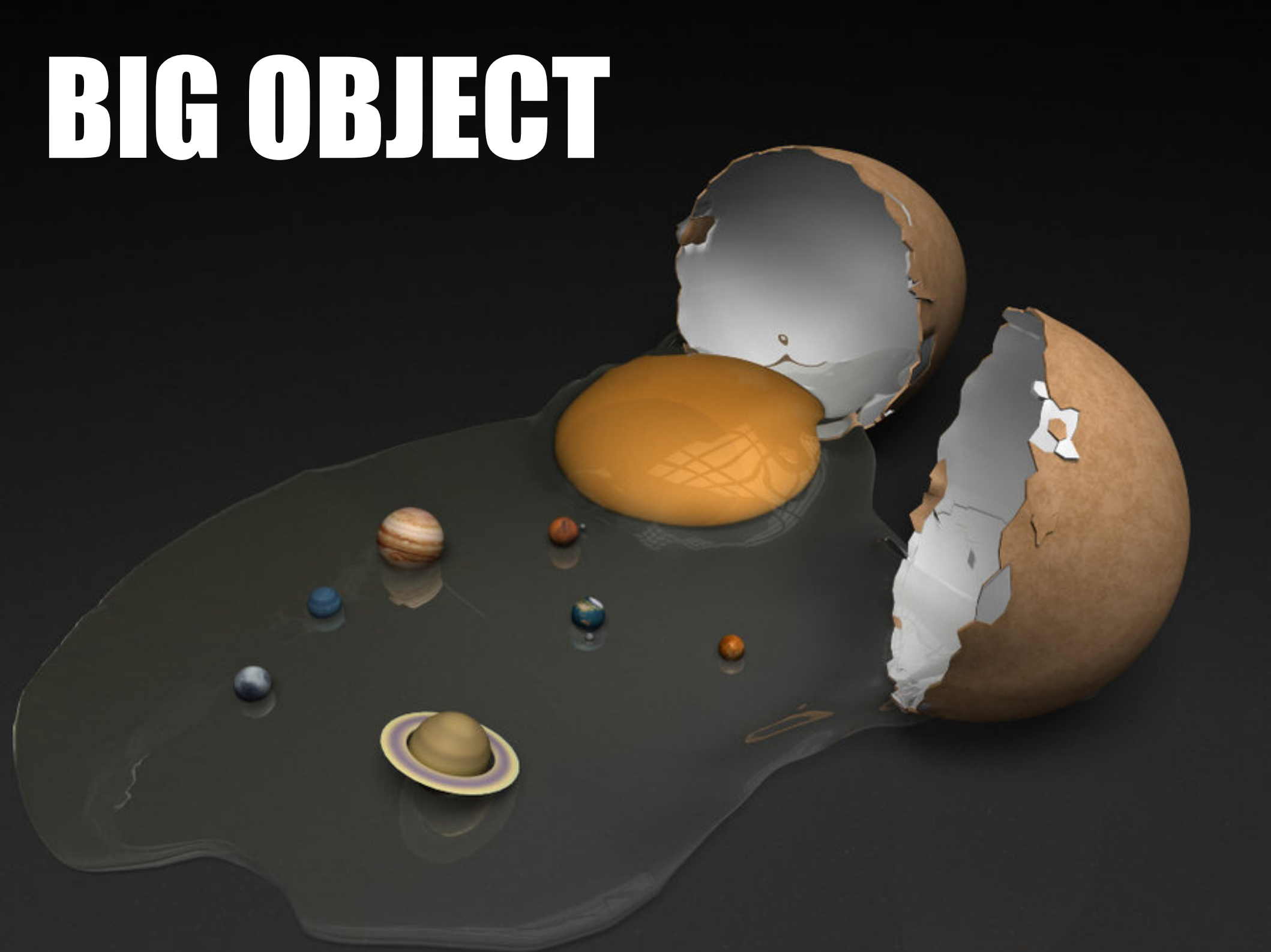
MERCURY

THE LAST PART

SHAMELESS

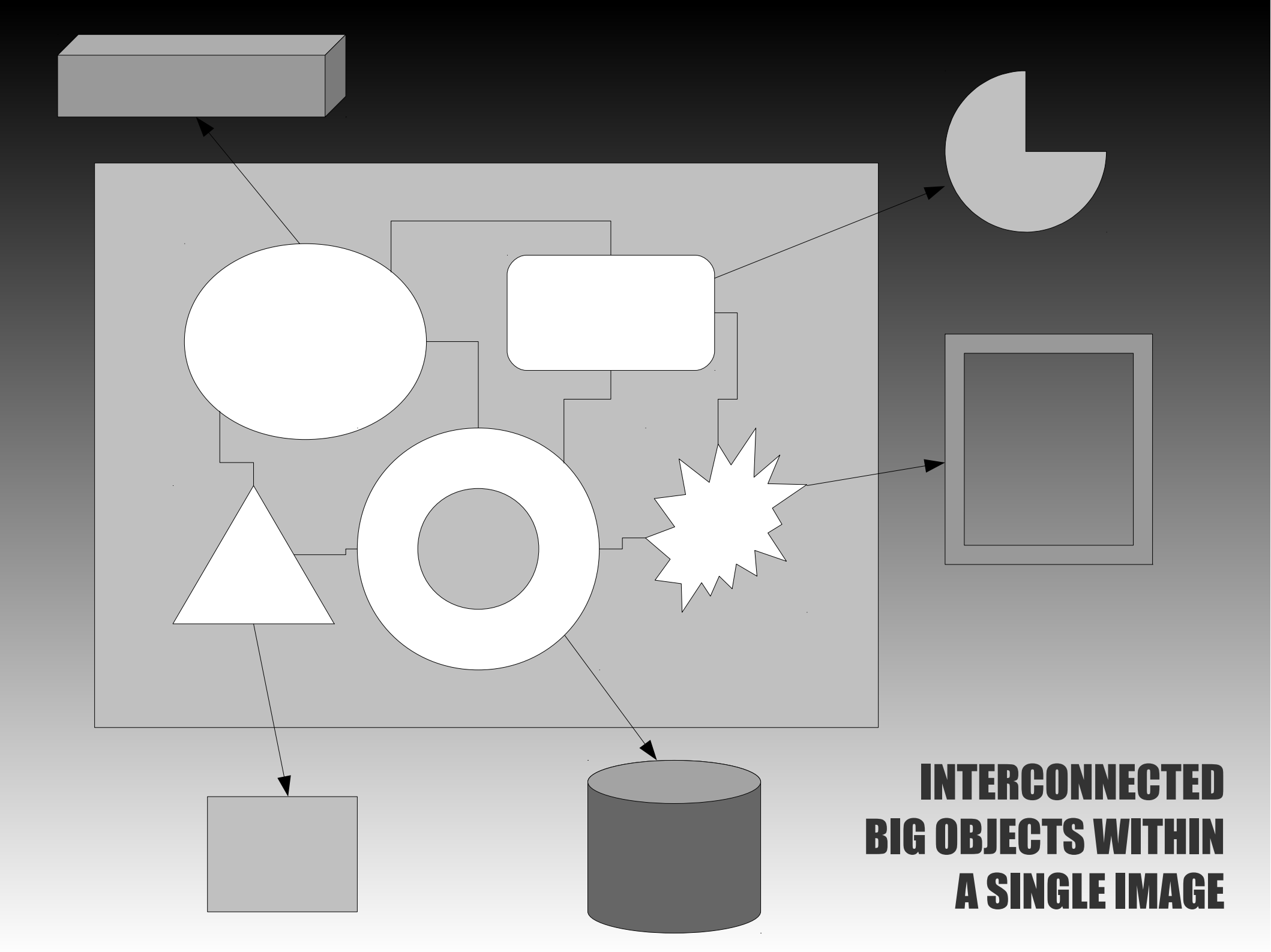
QUANTUM OBJECTS




BIG OBJECT

“One of the mistakes that we made years ago is that we made objects too small ..”

Alan Kay – “Programming and Scaling”


INTERCONNECTED
BIG OBJECTS WITHIN
A SINGLE IMAGE

PART ONE

DEATH TO SOCKETS

PART TWO

PROGRAM LOCALLY DEPLOY GLOBALLY

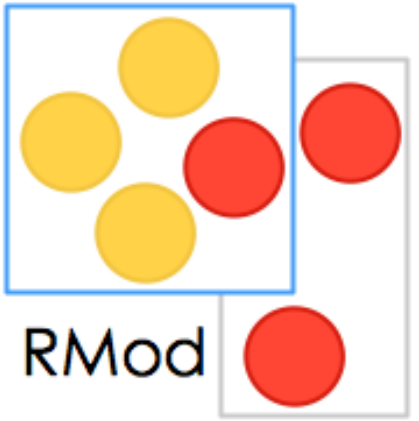
PART THREE

LET A THOUSAND SYSTEMS BLOOM

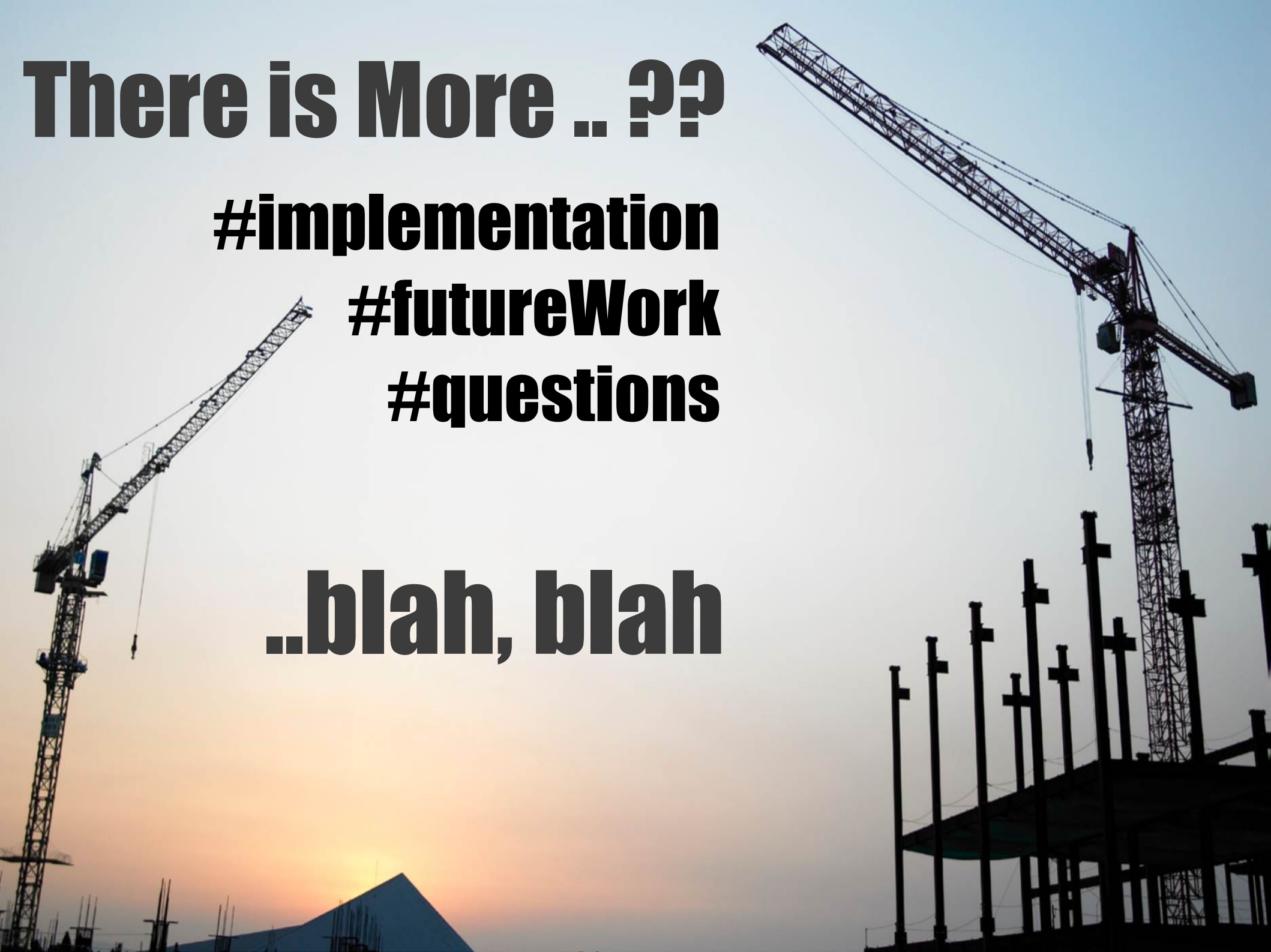
THE LAST PART

Seamless is
powered
by Pharo

calm asBomb



npapoylias@gmail.com


There is More .. ??
#implementation
#futureWork
#questions
..blah, blah